Software Requirements Specification

for

CS401 Final Project

Version 1.0

Prepared by

Group Name:

|  |  |  |
| --- | --- | --- |
| Luis Haro | Lhr004@gmail.com |  |
| Miguel Guzman |  |  |

|  |  |
| --- | --- |
| Instructor: | Christopher Smith |
| Course: | CS 401 |
| Date: | 2/21/2020 |

Contents

Contents ii

Revisions ii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 1

2 Overall Description 2

2.1 Product Overview 2

2.2 Product Functionality 2

2.3 Design and Implementation Constraints 2

2.4 Assumptions and Dependencies 3

3 Specific Requirements 4

3.1 External Interface Requirements 4

3.2 Functional Requirements 4

3.3 Use Case Model 5

4 Other Non-functional Requirements 6

4.1 Performance Requirements 6

4.2 Safety and Security Requirements 6

4.3 Software Quality Attributes 6

5 Other Requirements 6

Appendix A – Data Dictionary 7

Appendix B - Group Log 7

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| Draft Type and Number | Full Name | Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded. | 00/00/00 |

# 

# Introduction

## Document Purpose

This document will focus and, on the scope, requirements and specifications of the final project. The document will inform us of the principals of the project so students or instructor can have an idea of the whole project; this way, the members of the group can follow the requirement or make changes to it, in the same way other students can have a picture of the project and take over if it is necessary.

## Product Scope

This software project will create a new system to manage a gym. The software will allow employees and managers to manage every aspect a gym. User will be able to keep track of members, active or not, check condition of equipment, manage employees, add or remove equipment and members as well

## Intended Audience and Document Overview

This document is intended to be use for developers as a guide for requirements as that develop the software; in addition, it’s intended to inform users of what they can do with this new software. It can also be used as a guideline for students whom might have to work in this project in the future.

## Definitions, Acronyms and Abbreviations

UML - Unified Modeling language.

## Document Conventions

Use Arial font size 12 throughout the document for text. Use italics for comments. Document text should be single spaced and maintain the 1” margins found in this template. For Section and Subsection titles use size 14 Arial bold.

## References and Acknowledgments

Most of the information in this document can be found in out GitHub page:

https://github.com/luisHaroDev/finalProject.git

Template of this document can be found at CS.GMU.EDU

# Overall Description

## Product Overview

A close up of text on a white background

Description automatically generated

## Product Functionality

loadMembers: this function will allow the user to load information about members into a data structure for easy access and manage.

loadEquipment: this function will allow users to load information of equipment into a data structure of essay access and manage.

loadEmp: this function will allow users to load employee’s info into a data structure of essay access and manage.

addModifyMembers**:** Allow users to add new members into our data structure and modify information.

removeMembers: Allow users to remove members from the data base if needed.

checkMembership: Allow users to check if a member is an active member of the gym by their member ID.

checkEmp: will check if the employee is in the data base so he/she can access the software.

CheckStatusEquip: Allow user to check for equipment status.

removeEquip: will allow users to remove equipment as needed from the database.

addEquip: will allow the user to add equipment as needed to the database and place in a specific section of the gym.

reportDisplay: will give the user a full report of the gym equip status active and non-active members for marking purposes.

saveDatabase: will save all info change or added by the user to our main database file.

## Design and Implementation Constraints

We will be using and array of classes to store data about or employees, member and equipment. We will not have an array of 100 users, 50 equipment and 10 employees as max. our database will be 3 different txt file

## Assumptions and Dependencies

We will assume that our database is secure and not on site

# Specific Requirements

## External Interface Requirements

### User Interfaces

This software will be user friendly and will use GUI interface to interact will the user

### Hardware Interfaces

The only hardware interface will be a computer and IDE with the available to run Java programs such as Eclipse, NetBeans, etc.

### Software Interfaces

Our programs will interface with our databases and software will interface with different classes and components within the project.

## Functional Requirements

loadMembers: this function will allow the user to load information about members into a data structure for easy access and manage. We will use a .txt file to load info into a data structure, we shall create an object member and populated with the info in the .txt file

loadEquipment: this function will allow users to load information of equipment into a data structure of easy access and manage. We will use a .txt file to load info into a data structure, we shall create an object equipment and populated with the info in the .txt file

loadEmp: this function will allow users to load employee’s info into a data structure of easy access and manage. We will use a .txt file to load info into a data structure, we shall create an object employee and populated with the info in the .txt file

addModifyMembers**:** Allow users to add new members into our data structure and modify information. Function will take a name and add it to the data structure if he/she doesn’t not exist, but if the members exists it will update her/his info.

removeMembers: Allow users to remove members from the data base if needed. It will take a user ID and it will be removed from the data structure

checkMembership: Allow users to check if a member is an active member of the gym by their member ID. It will go to the data structure to check for a matching user ID if it is found it will give the information if the member is active or not. If it is not found it will also inform that.

checkEmp: will check if the employee is in the data base so he/she can access the software. It will check the data structure for employee info and allow access if found.

CheckStatusEquip: Allow user to check for equipment status. It will inform the user of equipment status such as active, need to be fix, reported as broken.

removeEquip: will allow users to remove equipment as needed from the database. it will check for a user and if the member exists, he/she will be removed from the data structure. If it is not found it will inform the user as well.

addEquip: will allow the user to add equipment as needed to the database and place in a specific section of the gym. Will take an equipment code and added it to data structure

reportDisplay: will give the user a full report of the gym equipment status active and non-active members for marking purposes. It will go to all 3 data structures (employee, equipment, members) and give a full report of their status and information.

saveDatabase: will save all info change or added by the user to our main database file. It will use the same .txt files we have used for information.

## Use Case Model

**CASE 1:**

1. User employee type his code.
2. The program along him/her to access the system.
3. The employee starts to check guess ID if they have access.
4. One customer wishes to become a member, so employee enter customer data into

The database.

1. Customer informs of a broken machine, employee sets that machine to inactive.
2. In the end of day, the employee saves all data.

**CASE 2:**

1. Manager ask for a report of all customer, employees and equipment
2. Employee log-in to the system.
3. The program check for his/her valid ID.
4. After validation the employee prints a report as requested

# Other Non-functional Requirements

## Performance Requirements

Our software will try to use our data structure as efficient as possible. We will delete user and add new one in the array in the same index to save memory. time complexity for search will be keep simple since it is a same data file.

## Safety and Security Requirements

Not much security features will be use for the exception if the user it is in the database.

## Software Quality Attributes

The program should be easy to use and the code easy to read in order to allow team members to understand, it should have comments have explain variable, methods and attributes so other student can follow if it is necessary.

# Other Requirements

Appendix A- Group Log